**Zigbee:**

Zigbee is a low power wireless technological standard created by Zigbee Alliance for WPANs based on IEEE 802.15.4 created for Control & Sensor Networks. It supports data rates up to 250kbps. It supports Star and Mesh topologies.

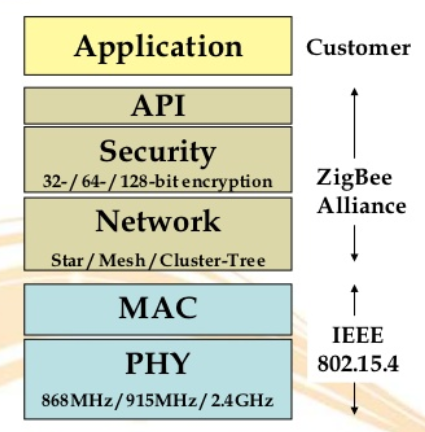
Zigbee consists of 2 types of devices –

* **Fully Functional Device (FFD):** It can communicate with other FFDs & RFDs and can function as a Coordinator, Router or End device.
* **Reduced Functional Devices (RFD):** It can communicate with only FFDs and can function only as an End device.

A Zigbee network consists of 3 components –

* **Coordinator:**
  + Initiates the network
  + Stores information about the network
  + Stores the routing algorithm
  + All devices communicate with it
  + Acts a bridge to other networks
  + WPAN must have exactly 1 FFD acting as Coordinator
* **Router:**
  + Optional component
  + Should be an FFD
  + Capable of extending network coverage area
  + Manages local address allocation & deallocation
* **End Device:**
  + Can either be an FFD or RFD according to the application
  + Optimized for low power applications
  + Cheapest component of the network
  + Communicates only with Coordinator or Router

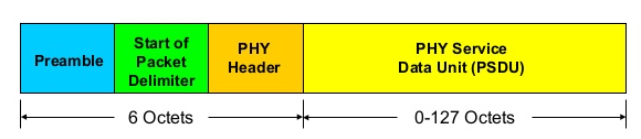
**Zigbee Stack:**



**Physical Layer:** The physical layer performs modulation on outgoing signals and demodulation on incoming signals. It transmits information and receives information from a source. The following table shows the physical layer frequency band, data rate, and channel numbers.

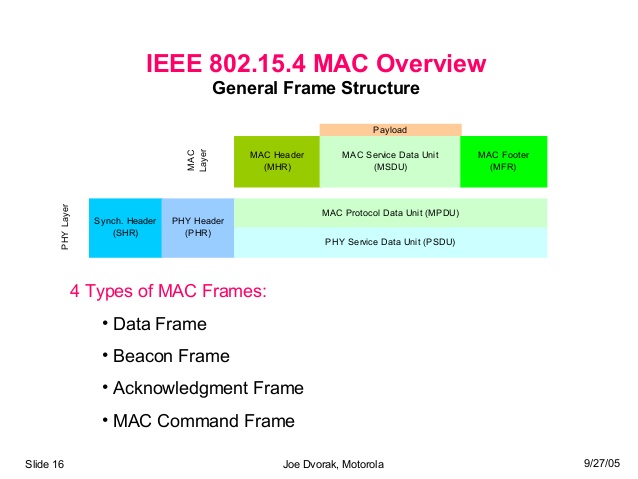
|  |  |  |  |
| --- | --- | --- | --- |
| Frequency Band | Country | Data Rate | Channel Numbers |
| 868.3MHz | European countries | 20Kbps | 0 |
| 902–928 MHz | United States | 40Kbps | 1–10 |
| 2.405GHz | Worldwide | 250Kbps | 11–26 |

The structure of a packet is as follows:

****

* Octets = Bytes
* Preamble – 32 bits
* Start of packet – 8 bits
* PHY header – 8 bits – stores PSDU length
* PSDU – 0 to 1016 bits – Data field

**MAC Layer:** It is responsible for performing functions like channel acquisition, frame security, error correction, specifying traffic mode, etc. The structure of a MAC layer packet is as follows:



Zigbee supports 4 types of frame format:

* Data frame – for normal data transmission
* Beacon frame – for handshakes
* Acknowledgment frame – for sending ACKs
* MAC command frame – for controlling digital pins of hardware XBee module.

**Traffic Types in Zigbee:**

* **Periodic:**
  + Sensor wakes up, checks for data & then goes to sleep
  + Application dictates the data rate
* **Intermittent:**
  + Application or some external stimulus determines the data rate (energy saver)
* **Repetitive:** 
  + Fixed data rate
  + Guaranteed Time Slots (GTS) are used to operate devices in fixed duration

**Network Layer:** The network layer is located between the MAC layer and application support sublayer. It provides the following functions:

* Starting a network
* Managing end devices joining or leaving a network
* Route discovery
* Neighbor discovery

**Application Support Layer:** It acts an interface between Network & Application Layers. It’s tasks include:

* Accepting data units from application layer
* Adding the header information of the frame to it
* Transferring the resulting frame to Network layer.

**Application Layer:** It is specified by the vendor. It consists of application objects which hold user applications & Zigbee device objects (ZDOs). Every device has descriptor called ‘ZigBee descriptor’. It contains following information:

* + Frequency band
  + Power description
  + Application flags
  + Application versions
  + Serial Number
  + Manufacturer

etc…

**Zigbee Device Object (ZDO):** It defines the role of the device in the network – Coordinator, Router or End Device. It also initializes API and Network layers and offers services like device/service discovery.

**Prototype:**

**Components used:**

* Arduino nano
* Xbee pro S2
* Arduino nano i/o breakout shield
* PIR motion sensor
* TTL serial camera
* Resistors
* SD card
* SD card reader
* Breadboard

**Architecture Diagram:**

<Put diagram here>

**Sequence Diagram:**

<Put diagram here>

* The network consists of 4 parts:
  + Motion Sensing & Detection (Router 1)
  + Central Actuator (Coordinator)
  + Camera (Router 2)
  + Lights (Router 3)
* All the router xbees are configured with config in AT mode & the coordinator is configured in API mode.
  + In AT mode, an XBee would broadcast any message it receives to its destination address (which is set to coordinator by default). Once the destination address is set, there is no provision for that XBee to send messages to any other destinations.
  + We do not configure the coordinator in AT mode as we don’t want it to broadcast every message it gets.
  + Also, API mode is suitable if the XBee is going to communicate with multiple XBees. This is because in every API packet, destination address has to be explicitly mentioned. So we can specify multiple addresses in our Arduino code & send different messages to their corresponding destination Xbees.

Let us explore the 3 parts of our network as mentioned above:

**Motion Sensing & Detection (Router 1):**

**Function:** Detects motion & sends a wireless message to the coordinator.

**Components:** PIR sensor, Arduino Nano, XBee module, Arduino Nano breakout shield.

**Architecture:**

* Fix the Arduino and Xbee in Arduino Nano breakout shield.
* The Out of PIR sensor is connected to Pin 2 of Arduino.
* Arduino Nano breakout board takes care of the connections between Arduino and XBee, so we don’t have to do anything explicitly.
* Upload the code in PIR\_Router.ino (\\Prototype code\arudino\PIR\_Router1) in Arduino.
* Upload the settings in Router\_AT\_profile.xpro (\\Prototype code\zigbee) in XBee to configure it as router for the network. The Pan ID is set to 1002 for the network.
* While uploading the code to Arduino, make sure you disconnect it from the nano board else it will throw errors.
* Ensure that you are correct info of the board in Tools menu. When using Arduino Nano board, make sure that Tools Menu> Boards > is set to Arduino Nano and Processor is set to Atmega 328P(old bootloader). Else it will throw out of sync error.

**Working:**

* When PIR sensor detects motion, Arduino writes a message into its Serial Port.
* To keep our message short, we have used “M” as our message payload. M here is our codeword for Motion Detected. The coordinator is programmed to look this message to trigger the camera.
* Since this router configured in AT mode, we don’t have to explicitly write every byte of the packet in Arduino’s Serial port. We are only are required to write the payload in Serial, the Xbee encodes other bytes from its internal profile settings.
* The Router XBee connected to the Arduino’s serial port, sends this message/packet wirelessly to the Coordinator. The size of this packet is 19 bytes.

Example packet: 7E 00 0F 90 00 13 A2 00 41 08 09 D7 8E 9B 01 4D 0D 0A 03

**Actuator (Coordinator):**

**Function:** Receives message from sensor & sends wireless messages to camera and light to turn them on/off.

**Components:** Arduino Nano, XBee module, Arduino Nano breakout shield.

**Architecture:**

* Fix the Arduino and Xbee in Arduino Nano breakout shield. (refer Motion Sensing & Detection part for details about breakout shield)
* Upload the code in Coordinator.ino (\\Prototype code\arudino\ Coordinator) in Arduino.
* Upload the settings in Coordinator\_API\_profile.xpro (\\Prototype code\zigbee) in XBee to configure it the coordinator for the network. The Pan ID is set to 1002.

**Working:**

* The Coordinator Xbee will receive message packet from router Xbee of sensing module and Arduino will read it through its serial port.
* If this message has our code (“M”) for detection of motion, a message will be sent to the camera to trigger.
* The Arduino will check & analyze this message to get the information out of it.
* It first checks for the Start delimiter of message i.e. 0x7E which is standard for all Zigbee packets.
* It reads the rest 18 packets and stores them in a byte array RFin\_bytes.
* It then looks if the payload of this packet has “M” in it. If so, it calls the activateCamera() function to trigger the camera.
* In activateCamera() function, we create an API packet for ‘Transmit Request’. Transmit Request is generally used to send a message to a destination.
* Refer the packet structure of API Transmit Request to understand the construction.
* Arduino will construct this packet by writing it byte by byte in its Serial port.
* The XBee will transmit this message from its serial port.
* Every byte encoding in the code has comments with it indicating the part of the packet that byte corresponds to.
* Here we encode the letter “A” into the payload of the message packet denoting ‘Activate camera’.
* Notes: The 3rd byte written is the length of the message. The length of the message is calculated by counting everything in the packet after this 3rd byte excluding the checksum. In the end checksum is calculated, it a sum of every non-zero byte after the length byte of the packet.
* Check sum marks the last byte of the message packet sent.
* Coordinator calls one more function right after activateCamera() i.e. activateLight()
* activateLight()function wirelessly turns on the LEDs. It is a remote AT command to control Router 3 Xbee from Coordinator.
* We control D3 of Router 3 XBee, to specify pin number as D3 in the packet structure we use the hex converted values 0x44 0x33 while writing bytes in Arduino’s serial port.
* To set pin D3 of Xbee to HIGH, write 0x5 as parameter value, to reset it to LOW write 0x4 as parameter value in the packet.

**Camera (Router 2):**

**Function:** Receives message from the coordinator, clicks a picture & stores in the SD card.

**Components:** TTL Serial Camera, Arduino Nano, XBee module, Arduino Nano breakout shield, Sd Card, Sd Card reader, Breadboard, Resistors

**Architecture:**

* Fix the Arduino and Xbee in Arduino Nano breakout shield. (refer Motion Sensing & Detection part for details about breakout shield)
* For pin connections of Arduino, Camera and SD card reader, please refer: <https://learn.adafruit.com/ttl-serial-camera/arduino-usage>
* Upload the code in Camera\_Router2.ino (\\Prototype code\arudino\ Camera\_Router2) in Arduino.
* Upload the settings in Router\_AT\_profile.xpro (\\Prototype code\zigbee) in XBee to configure it as router for the network. The Pan ID is set to 1002 for the network.

**Working:**

* The Xbee receives the message packet from the coordinator.
* Since this Xbee is configured in AT mode, it directly extracts the payload of the message packet and writes it in Serial port of the Arduino.
* The arudino checks if the payload is equal to “A” (the code for activating the camera), if so it clicks a picture from the camera and stores it in the SD card.
* We can change the resolution of the image in the code.
* It takes around xx secs to store an image of approx. 50kb.
* Once the pic is captured, it displays a message “Picture captured & stored in SD card”,

**Messages:**

M – **M**otion detected – Sensor to Coordinator

A – **A**ctivate camera – Coordinator to Camera

D – Capturing image **D**one (Positive ack) – Camera to Coordinator

F – Capturing image **F**ailed (Negative ack) – Camera to Coordinator

R – **R**esume sensing – Coordinator to Sensor

**Imp Note : Make sure that Arduino is not writing these codewords in any other Serial.print() or Serial.write() command. It would break the logic as the receiving Xbee would get a false alarm that this message is generated by the Arduino. One trick is to use any uppercase letters only for codewords & lowercase for any other messages that you want Arduino to print in its Serial port.**

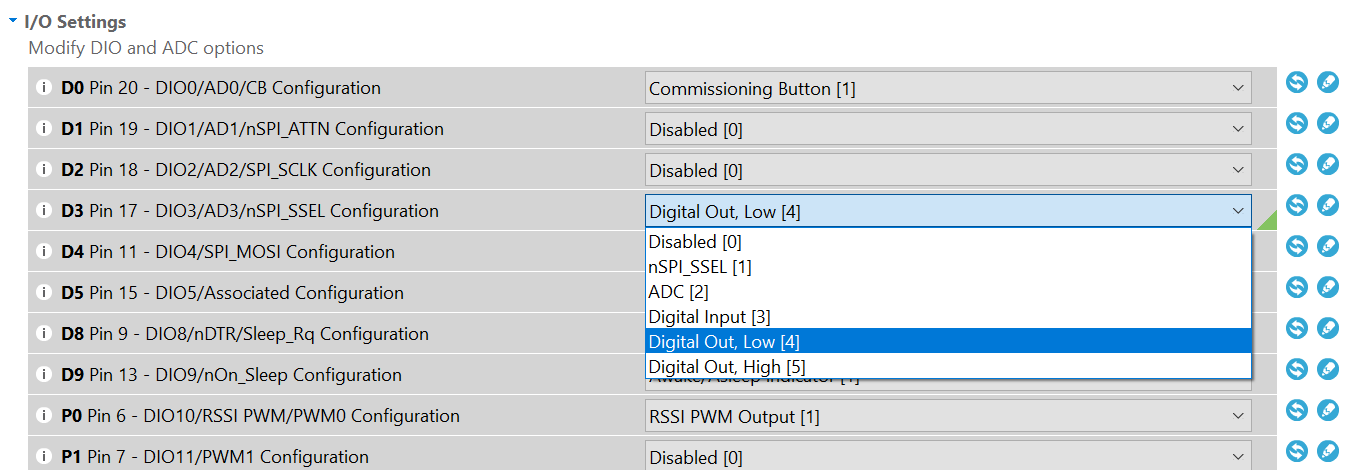
**Lights (Router 3):**

**Function:** Receives message from the coordinator and turns on the LEDs.

**Components:** XBee module, XBee breakout board, Breadboard, LEDs, connecting wires

**Architecture:**

* Upload the settings in Router\_AT\_profile.xpro (\\Prototype code\zigbee) in XBee to configure it as router for the network. The Pan ID is set to 1002 for the network.
* In addition to the above settings, set D3 pin of the XBee to Digital Low[4]. This step is necessary as we want to use this pin to control the LEDs.



* Connect the xbee on breadboard & supply 3.3V to it. Take a connection out from D3 & connect it to LEDs.

**Working:**

* The Xbee receives the message packet from the coordinator.
* Since this Xbee is configured in AT mode and when it receives a message in Remote AT Command frame format, it decodes the pin number and the value to which it must be set.
* Initially, D3 is configured to be LOW. Once the coordinator gets a message that motion has been detected, it will send a message which will set D3 to HIGH which turns on the LEDs.
* Then, when the camera is done capturing the image and it sends and ACK back to Coordinator, then the Coordinator again sends a message to reset D3 to LOW which turns off the LEDs.

**Scope for improvement:**

* The Arduino takes a lot of time to capture a picture & store it in the SD card.
* This is because the serial port of Arduino is very slow.
* Since the image capture is so slow, this cannot be enhanced to have live video streaming capabilities.
* The solution is to use a Raspberry Pi instead of an Arduino. Pi has greater significantly higher computational power than Arduino & it can support video live streaming with good response time.

**Enhanced Version:**

**Camera (Router 2):**

**Function:** Receives message from the coordinator, clicks pictures and then starts live video streaming on a webpage.

**Components:** Rasberry Pi, Arduino Uno, XBee module, Camera<model>, Sd Card.

**Architecture:**

* The XBee module is connected to the Arduino.(Rx to Tx & vice-vera)
* The Rasberry Pi is connected to the Arduino (using a USB cable) and a camera.
* Upload the code in Camera\_Pi\_Router2.ino (\\Enhanced Pi Camera code\arduino\ Camera\_Pi\_Router2) in Arduino.
* Upload the settings in Router\_AT\_profile.xpro (\\Prototype code\zigbee) in XBee to configure it as router for the network. The Pan ID is set to 1002 for the network.

**Working:**

* The Xbee receives the message packet from the coordinator & Arduino reads it from the serial port. If it receives a packet with payload ‘A’, the arduino writes ‘A’ in its serial port.
* The Rasberry Pi is constantly reading Arduino’s serial port. When it reads an ‘A’ it triggers the camera function.
* The arduino is programmed to timeout after 30 seconds, once this interval is over, it kills the camera recording and sends a message with payload “D” back to the coordinator. When the coordinator receives this message, it turns off the lights and sends resume sensing message to the PIR sensor to restart the sensing operation.